```
package com.analyticsanvil.custominputformat;

import java.io.BufferedReader;
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import java.util.regex.PatternSyntaxException;

public class ReadASCIIGridFile {

    public static byte[] translateInput(byte[] inputBytes) {

        int ncols = -1;
        int nrows = -1;
        String pattern;
        Pattern r;
        Matcher m;
        String input, outputTmp;
        StringBuilder output = new StringBuilder();
        double xllcorner, yllcorner, cellsize;
        xllcorner = 0.0d;
        yllcorner = 0.0d;
        cellsize = 0.0d;
        int x = 0, y = 0;
        String[] readingsArray = null;
        InputStream is = null;

        try {
            is = new ByteArrayInputStream(inputBytes);
            BufferedReader br = new BufferedReader(new InputStreamReader(is));

            // Read NCOLS from header
            input = br.readLine();
            pattern = "NCOLS (\\d+)";
            r = Pattern.compile(pattern);
            m = r.matcher(input);
            if (m.find()) {
                ncols = Integer.parseInt(m.group(1));
                input = br.readLine();
            }

            // Read NROWS from header
            pattern = "NROWS (\\d+)";
            r = Pattern.compile(pattern);
            m = r.matcher(input);
            if (m.find()) {
                nrows = Integer.parseInt(m.group(1));
                input = br.readLine();
            }
```

```
// Read starting longitude from header
pattern = "XLLCORNER (-?\\d+.\\d+)";
r = Pattern.compile(pattern);
m = r.matcher(input);
if (m.find()) {
    xllcorner = Double.parseDouble(m.group(1));
    input = br.readLine();
}

// Read starting latitude from header
pattern = "YLLCORNER (-?\\d+.\\d+)";
r = Pattern.compile(pattern);
m = r.matcher(input);
if (m.find()) {
    yllcorner = Double.parseDouble(m.group(1));
    input = br.readLine();
}

// Read step size from header (i.e. how much to increment lat / long for each row / col)
pattern = "CELLSIZE (\\d+.\\d+)";
r = Pattern.compile(pattern);
m = r.matcher(input);
if (m.find()) {
    cellsize = Double.parseDouble(m.group(1));
    input = br.readLine();
}

// Read data section of file
pattern = "-?(\\d+) .*";
r = Pattern.compile(pattern);

// Keep reading through the file until the end is reached (no more data)
while ((input = br.readLine()) != null) {
    m = r.matcher(input);
    if (m.find()) {

        try {
            // Split readings on whitespace
            readingsArray = input.split("\\s+");
        } catch (PatternSyntaxException ex) {
            //
        }

        // Read all columns on this row (according to the nubmer of columns declared into the file header)
        for (x = 0; x < ncols; x++) {

                        // Output a row of data (separated by tab character):
            // Southern Latitude, Western Longitude, Northern Latitude, Eastern Longitude, Reading
            outputTmp = String.format("%f\t%f\t%f\t%f\t%s\n", y * cellsize
                    + yllcorner, x * cellsize + xllcorner, (y + 1)
                    * cellsize + yllcorner, (x + 1) * cellsize
                    + xllcorner, readingsArray[x]);
            // Add this row to the output
            output.append(outputTmp);
```

```
                }
                // Increment y-coordinate (i.e. latitude)
                y++;
            }
        }

    } catch (IOException io) {
        io.printStackTrace();
    }

    // Output all rows of the processed data
    return output.toString().getBytes();

  }

}
```